

Enhancing your Existing RPG Applications with PHP

Delivering files from the IFS to the browser
from your 5250 application.

Brian May
IBM i Modernization Specialist
Profound Logic Software

Webmaster and Coordinator
Young i Professionals

Overview

- Intro to PHP in IBM i
- Writing a Simple PHP Script (Hello World in PHP)
- A Few Really Useful PHP Functions
- In Depth Look at the “GetFile.php” Script
- Securing the “GetFile.php” Script
- Calling the “GetFile.php” Script via Browser from RPG

PHP on IBM i

What is it? How do I get it? Why should I?

PHP on IBM i: What is it?

- PHP is a programming language.
- According to PHP.net:
“PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.”
- PHP can be imbedded within HTML.
- PHP can be both Object Oriented and Procedural.

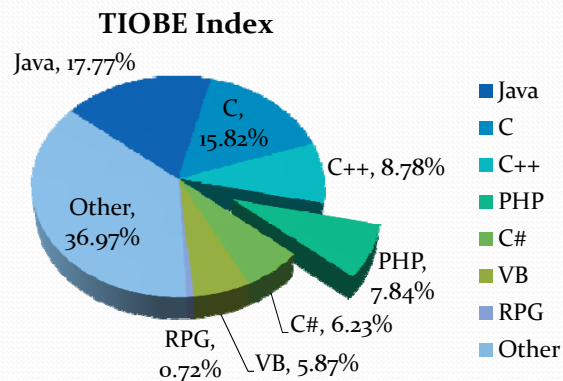
PHP on IBM i: How do I get it?



- Zend
 - <http://www.zend.com/en/products/server/zend-server-ibm-i>
 - Two install methods available
 - SAVF
 - Automated Installer (Windows .EXE)

PHP on IBM i: Why Should I?

- PHP is one of the most popular programming languages in the world



PHP on IBM i: Why Should I?

- It's FREE!!!
- PHP is relatively easy for RPG'ers to understand.
 - Supports procedural programming
 - Easy bridge into OO
- It's FREE!!!
- When coupled with MySQL, it opens the IBM i to run any of the thousands of open source applications written for the LAMP stack (Linux, Apache, MySQL, PHP)
- Did I mention that it's free?
- PHP can directly access and use any native objects on IBM i including Programs, Service Program Procedures, Data Areas, Data Queues, System Values, Spool Files, Jobs, User Spaces, and just about anything else.

Writing a Simple PHP Script

Overview of the structure of PHP

“Hello World” – PHP Style

```

1 <?php
2     // Call the error function to test the script
3     error('This is a test of the error function.');
```

4

```

5 function error($message)
6 {     // Very Simple HTML with the message embedded
7     ?>
8     <html>
9         <head>
10            <title><?php echo $message?></title>
11        </head>
12        <body>
13            <h1><?php echo $message?></h1>
14        </body>
15    </html>
16 <?php
17     exit;
18 }
19 ?>
```

Elements of the PHP Script

- All sections of PHP code must begin with the **<?php** tag and end with the **?>** tag.
- PHP statements end in semicolons. (Just like RPG.)

```

1 <?php
2     // Call the error function to test the script
3     error('This is a test of the error function.');
```

4

```

5 function error($message)
6 {     // Very Simple HTML with the message embedded
7     ?>
8     <html>
9         <head>
10            <title><?php echo $message?></title>
11        </head>
```

Elements of the PHP Script

- Functions are defined by the keyword **function**.
- Parameters are defined in parenthesis after the function name.
- The function's logic is enclosed in curly braces **{}**.

```
5 function error($message)
6 { // Very Simple HTML with the message embedded
7     ?>
8     <html>
9     <head>
10        <title><?php echo $message?></title>
11    </head>
12    <body>
13        <h1><?php echo $message?></h1>
14    </body>
15 </html>
16 <?php
17     exit;
18 }
19 ?>
```

Output of the Error() function



Useful PHP Functions

PHP functions that will be used in the GetFile.php example.

Useful PHP Functions

- `file_exists()`
 - This function takes in absolute file path as a parameter and returns true if the file exists.
 - Example:

```
$found = file_exists("/myfolder/images/abc.jpg") ;
```
- `header()`
 - This function outputs a raw HTTP header.
 - Example

```
header("Status: 404 Not Found");
```

Useful PHP Functions

- `readfile()`
 - This function reads in the content of the file specified as a parameter and writes it directly to the output buffer.
 - Useful for streaming streaming files (PDF, JPG, etc.)
 - Example:
`readfile("/tmp/report.pdf") ;`
- `fopen()`, `fgets()`, `fgetss()`, `fclose`
 - Functions to use stream files
 - More on this later

Examining GetFile.php

Simplified version of script to retrieve any file from the IFS

GetFile.php – Unsecured Version

```
<?php
// If the above is not the FIRST line in the file, it WON'T WORK!

// Get variables passed in
$directory = $_REQUEST['dir'];
$fileName = $_REQUEST['file'];

// make sure file exists
if (!file_exists("$directory/$fileName"))
{
    error("File $directory/$fileName does not exist.");
}

// Tell the browser to open file as an attachment
header("Content-Disposition: attachment; filename=$fileName");
header("Content-Type: application/octet-stream");

// Stream file to browser
readfile("$directory/$fileName");

// Leave the script explicitly
exit;
```

GetFile.php - Continued

```
function error($message)
{
    // Very Simple HTML with the message embedded
    ?>
    <html>
        <head>
            <title><?php echo $message?></title>
        </head>
        <body>
            <h1><?php echo $message?></h1>
        </body>
    </html>
    <?php
        exit;
    }
    ?>
```

Getting Input

- This script is expecting parameters to be passed within the URL. So the script is called using
http://your_i:10088/GetFile.php?&dir=/folder&file=file.txt
- The \$_REQUEST array is automatically created so we can get to the parameters passed.

```
// Get variables passed in
$directory = $_REQUEST['dir'];
$fileName = $_REQUEST['file'];
```

Checking For the File

- ! before file_exists() is a logical NOT.
- Directory and file name embedded in error message.

```
// make sure file exists
if (!file_exists("$directory/$fileName"))
{
    error("File $directory/$fileName does not exist.");
}
```

Outputting to the Browser

- HTTP headers describe the output
 - Output is an attachment
 - Output is an octet-stream
- Send the file to the output buffer and GET OUT.

```
// Tell the browser to open file as an attachment
header("Content-Disposition: attachment; filename=$fileName");
header("Content-Type: application/octet-stream");

// Stream file to browser
readfile("$directory/$fileName");

// Leave the script explicitly
exit;
```

Demo

Making the Script Secure

Accepting the directory and file name in the URL is not a good idea.

What's Wrong with GetFile.php

- Passing the directory and file name in the URL allows a user to access any folder on the box.
- What if someone asked for directory /QSYS.LIB/PAYROLL.LIB/EMPLOYEE.FILE and file EMPRECORD.MBR ?

GetFile.php – Secured Version

```
<?php
// If the above is not the FIRST line in the file, it WON'T WORK!

// Retrieve input parameter
$id = $_REQUEST['id'];

// Use id to retrieve directory and file
get_parm_file($id, $directory, $fileName);

// make sure file exists
if (!file_exists("$directory/$fileName"))
{
    error("File $directory/$fileName does not exist.");
}

// Tell the browser to open file as an attachment
header("Content-Disposition: attachment; filename=$fileName");
header("Content-Type: application/octet-stream");

// Stream file to browser
readfile("$directory/$fileName");

// Leave the script explicitly
exit;
```

GetFile.php - Continued

```
function error($message)
{
    // Very Simple HTML with the message embedded
    ?>
    <html>
        <head>
            <title><?php echo $message?></title>
        </head>
        <body>
            <h1><?php echo $message?></h1>
        </body>
    </html>
    <?php
        exit;
    }
}

function get_parm_file($inID, &$outDir, &$outFile)
{
    // get file information from temp file

    // Open temp file
    $handle = fopen("/tmp/$inID.tmp", "r");
```

GetFile.php - Continued

```

if ($handle)
{
    // Get file information
    $outDir = trim(fgets($handle));
    $outFile = trim(fgets($handle));
}
else
{
    error("Unable to retrieve ID $inID.");
}

// Close temp file
fclose($handle);

// Delete the parm file
unlink("/tmp/$inID.tmp");
}
?>

```

Modifications to GetFile.php

- Directory and file are no longer passed via URL. Instead, an ID string is passed as the parameter.
- New function `get_parm_file()` created to retrieve directory and file from a temp file on the IFS.

```

// Retrieve input parameter
$id = $_REQUEST['id'];

// Use id to retrieve directory and file
get_parm_file($id, $directory, $fileName);

```

get_parm_file() Function

- The function has 3 parameters
 - \$inID – the ID string passed in on URL
 - \$outDir – the directory passed back to the caller
 - \$outFile – the file name passed back to the caller
- fopen() used to open the temp file

```
function get_parm_file($inID, &$outDir, &$outFile)
{
    // get file information from temp file

    // Open temp file
    $handle = fopen("/tmp/$inID.tmp", "r");
```

get_parm_file() Function

- Check \$handle to ensure the temp file was opened
- Use trim() and fgets() to retrieve data from temp file

```
if ($handle)
{
    // Get file information
    $outDir = trim(fgets($handle));
    $outFile = trim(fgets($handle));
}
else
{
    error("Unable to retrieve ID $inID.");
}
```

get_parm_file() Function

- Close the temp file using fclose()
- Delete the temp file using unlink()

```
// Close temp file
fclose($handle);

// Delete the parm file
unlink("/tmp/$inID.tmp");
}
```

Format of Temp File

- The temp file is a simple text file with a .tmp extension
- The first line of the text file should contain the directory
- The second line of the file should contain the file name.

Demo

Integrating With Existing Applications

That's great, but how does my RPG program use it?

Required Programs

- We will look at two programs used in calling GetFile.php from your native applications
- GFBUILDURL – RPG program that creates the parm file in the IFS and generates the URL, including a random ID string.
- RUNBROWSER – CL program to invoke the browser on the users desktop.
- I highly recommend these processes be deployed as subprocedures.

GFBUILDURL Program

- Program begins by defining variables and constants needed

```
hOption(*NoDebugIO:*SrcStmt) BndDir('QC2LE') DftActGrp(*No) ActGrp('BRIAN')

// Variable Definiitons

d Seed          s          10i 0 Inz(0)
d Temp_File_Name s          15
d Temp_File      s          256
d Temp_File_Handle...
d               s          10i 0
d Mode           s          10i 0 Inz(511)
d Path           s          Like(Path_to_File)
d File           s          Like(File_Name)

// Constants

d System_URL     c          'http://www.youngipprofessionals+
d               .com:89'
```

GFBUILDURL (Continued)

- Lots of APIs defined to work with stream files (temp file)

```

d O_WRONLY      c      2
d O_CREAT       c      8
d O_CODEPAGE    c      8388608
d O_TEXTDATA    c      16777216

d CRLF          c      Const(x'0d25')

// External Procedures

// API call to open a stream file

d file_open      PR      10I 0 ExtProc('open')
d path          *      Value Options(*String)
d oflag          10I 0 Value
d mode           10U 0 Value Options(*NoPass)
d codepage       10U 0 Value Options(*NoPass)

```

GFBUILDURL (Continued)

- Also an API to generate random numbers

```

// API call to write data to a stream file

d File_write      PR      10I 0 ExtProc('write')
d fildes          10I 0 Value
d buf             *      Value
d nbyte           10U 0 Value

// API call to close a stream file

d File_close      PR      10I 0 ExtProc('close')
d fildes          10I 0 Value

// Generate a random #

d Random          PR      ExtProc('CEERANO')
d Seed            10i 0
d Random_Float    8f
d Feedback_Cd     12

```

GFBUILDURL (Continued)

- Definition of internal procedure and program parameters

```
// Internal Procedures

d Generate_Rndm_Name...
d                                     pr                                     15

// Entry Parmes

d GFBUILDURL          pr                                     ExtPgm('GFBUILDURL')
d   Path_to_File      256                                     Const
d   File_Name         256                                     Const
d   URL               128

d GFBUILDURL          pi
d   Path_to_File      256                                     Const
d   File_Name         256                                     Const
d   URL              128
```

GFBUILDURL (Continued)

- Main program logic starts by generating the ID
- Creates and opens the temp file using the ID

```
/Free
// Mainline Code

// Set up temp file name and location
Temp_File_Name = Generate_Rndm_Name() ;
Temp_File = '/tmp/' + Temp_File_Name + '.tmp' ;

// Create temp file
Temp_File_Handle = File_Open(%Trim(Temp_File):
                           O_CREAT+O_WRONLY+O_CODEPAGE: Mode : 819) ;

// Close and reopen temp file (makes conversion to ASCII automatic)
File_Close(Temp_File_Handle) ;
Temp_File_Handle = File_Open(%Trim(Temp_File) : O_WRONLY+O_TEXTDATA) ;
```

GFBUILDURL (Continued)

- Outputs the directory and file name to temp file
- Closes file

```
// Write directory to first line of temp file
Path = %Trim(Path_to_File) + CRLF ;
File_Write(Temp_File_Handle : %Addr(Path) :
           %Len(%Trimr(Path))) ;

// Write file name to next line of temp file
File = %Trim(File_Name) + CRLF ;
File_Write(Temp_File_Handle : %Addr(File) :
           %Len(%Trimr(File))) ;

// Close temp file
File_Close(Temp_File_Handle) ;
```

GFBUILDURL (Continued)

- Builds and returns the URL with the random ID to the calling program

```
// Build the URL to pass back to caller
URL = System_URL + '/getfile.php?&id=' + %Trim(Temp_File_Name) ;

*inLR = *On ;
Return ;

/End-Free
```

GFBUILDURL (Continued)

- Subprocedure used to generate random ID

```
//-----//
// Subprocedures //
//-----//

//-----//
// Procedure: Generate_Rndm_Name //
// Procedure to generate a random name for the temp file used //
// to pass parameters to the getfile php program. //
//-----//

p Generate_Rndm_Name...
p          B

d Generate_Rndm_Name...
d          pi          15
```

GFBUILDURL (Continued)

- Fields local to the subprocedure
- Valid_Chars should contain any character allowed in the ID

```
//-----//
// Local Definitions //
//-----//

d Random_Float      s          8f
d Feedback          s          12

d Valid_Chars       c          'ABCDEFGHIJKLMNOPQRSTUVWXYZ+
d                   d          0123456789'

d Counter           s          3i 0
d Random_String     s          15  Inz(*Blanks)
```

GFBUILDURL (Continued)

- For loop controls the number of characters in the ID
- Random # between zero and one generated
- Random fraction used to substring out a character from Valid_Chars

```

/Free

//I want a 15 character ID
For Counter = 1 to 15 ;
    //Get the random number
    Random(Seed:Random_Float:Feedback) ;

    // Use random number to set a random character in ID
    %SubSt(Random_String:Counter:1) =
        %SubSt(Valid_Chars:%int(Random_Float*%len(Valid_Chars)+1):1) ;
EndFor;

```

GFBUILDURL (Continued)

- Subprocedure returns the new ID

```

Return Random_String ;

/End-Free

p Generate_Rndm_Name...
p

```

RUNBROWSER

- STRPCO command starts the “PC Organizer” service if not already started.
- STRPCCMD command actually instructs the user’s PC to start the browser and open the URL

```
Pgm          Parm(&URL)

Dcl          Var(&URL) Type(*Char) Len(128)

StrPCO       PCTA(*No)
MonMsg       MsgID(CPF0000)

StrPCCmd     PCCmd('rundll32 url.dll,FileProtocolHandler ' || +
                  &URL) Pause(*No)

EndPgm
```

Demo

Questions?

Thank You!

About the Author

Brian May is an IBM i Modernization Specialist for Profound Logic Software. He is also webmaster and coordinator for the Young i Professionals (<http://www.youngiprofessionals.com>). He is a husband and father of two beautiful girls. Brian can be reached at bmay@profoundlogic.com

